SEMINAR REPORT

高速ストリーム暗号ＫＣｉｐｈｅｒ-２の研究開発

（株）KDDI 総合研究所
主席特別研究員
兼 KDDI（株）　技術企画本部
情報セキュリティフェロー
田中　俊昭 氏

ただいま御紹介に預かりましたＫＤＤＩ総合研究所の田中と申します。本日はこのような機会をいただきまして誠にありがとうございます。タイトルは「高速ストリーム暗号ＫＣｉｐｈｅｒ-２の研究開発」ということで講演をさせていただきます。

こちらは先ほど御紹介いただきました今年１月にＳＣＡＴ会長賞を受賞いたしましたものに関するお話になります。この受賞につきましては私だけではなく４名の受賞者がおります。九州大学の櫻井先生、弊社の清本、仲野の４名で受賞させていただきました。

私のお話は暗号技術ということで少し固めのお話になりまして、これを１時間話すとたぶんお聞きになっている方も大変だと思いますが、お付き合いいただければと思います。今日の内容は、当然、暗号ということを御存じの方もいらっしゃいますけれども、あまり御存じではない方もいらっしゃるということで、まず最初に暗号技術の概要を説明させていただいて、その後、今回受賞になりましたＫＣｉｐｈｅｒ-２の研究開発ということになります。こちらのＫＣｉｐｈｅｒ-２は約 10 年前に一通りプロジェクトも終わり、それから皆さんに使っていただいてということで、いろいろと商用展開をやっていることになります。これだけでは少し前の研究開発ということになりますので、最後に暗号技術の今後ということで今、どういった暗号技術が注目されているかというところも少し、受賞の枠を外れますけれども補足で説明をさせていただきたいと思います。

私は入社してほぼずっと研究所におりまして、もう 30 年以上たちますけれども、最初はまだセキュリティーの研究は世の中にはあまりなくて、やられている方も限られていました。やはりセキュリティーが非常に重要になってきましたのは約 25 年前でしょうか。インターネットが学術ネットワークから商用ネットワークに展開されてきたのは 1995 年ぐらいだと思いますが、ちょうど Windows95 も出て、世の中にＩＣＴの技術が広まっていったという中で、やはりセキュリティーの問題が顕在化してきたと思っていまして、そのあたりから軸足をセキュリティーの研究に置いて進めております。

暗号化技術概要

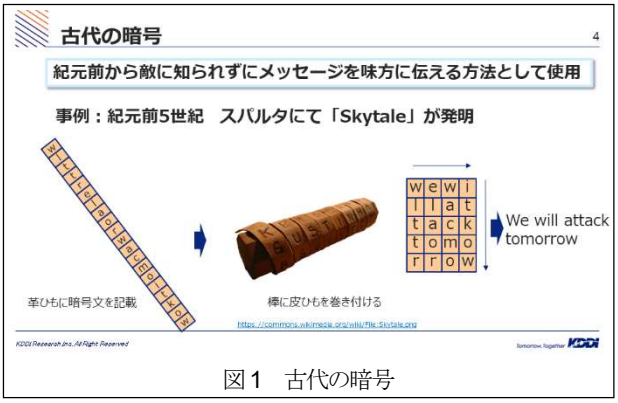

図１　古代の暗号

暗号は御存じのように古くからあります。紀元前からいわゆる戦争で敵に電文を知られないように味方に伝える方法ということで、従来から使われてきております。例えばということで１つ例を挙げさせていただきますと、紀元前５世紀、スパルタで「Skytale」、スキュタレーと読むらしいのですが、こういう物が使われていたという記録が残っています。(図 1)

こちらにありますように皮ひもに縦に字が書いてあります。これだけでは何が書いてあるか分かりませんけれども、これを棒にクルクルと巻き付けて横に読むと、こういう感じで文字が読めるようになります。例えば「We will attack tomorrow（明日、攻撃するぞ）」ということが浮き出てくるといった、こういう暗号が従来は使われていました。

その他有名なものとしてはローマ時代にシーザー暗号といったものもありますし、ずっと暗号は使われてきていまして、第二次世界大戦時のドイツではエニグマという有名な暗号が使われており、これは敵に解読されていた事実もございます。

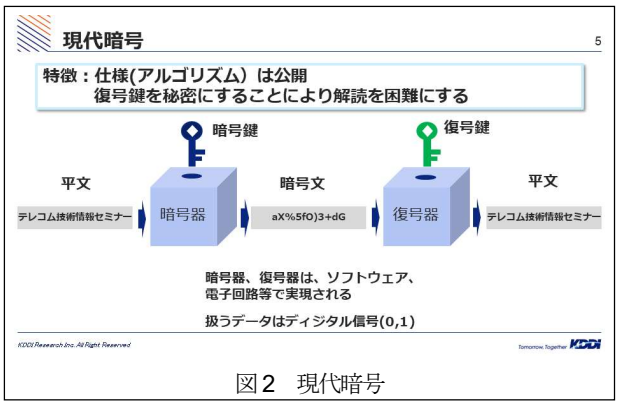

図２　現代暗号

そのような形で暗号は非常に歴史のあるものですけれども、特に戦後、現代暗号というものができました。こちらは最近のコンピュータ等の発達によっていろいろな処理ができるようになったということで、これを暗号に応用しようということが流れとして出てきました。

今の暗号はどういうふうにデータを暗号化しているかという仕様は公開します。ただ、そこには鍵というものがあって、特に復号する鍵を秘密にすることで安全性を保つという技術になります。

この絵(図2)で言うと、例えば平文がありまして、これが暗号器にかかって鍵を設定すると暗号文が出てきます。今度は復号器でまた復号鍵を入れると平文に戻りますという仕組みです。当然、最近はコンピュータや電子回路が発達してきましてこういう物を使っておりますので、基本的には暗号復号器はソフトウェアあるいは電子回路で実現されております。ここで扱えるデータは平文という文字を書いていますが、基本的には０、１のデジタルデータになっています。

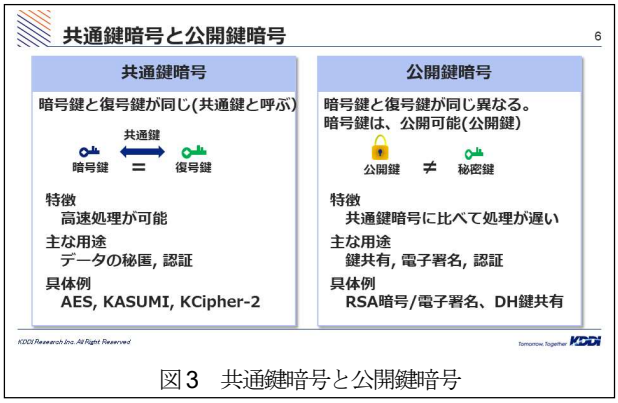

図３　共通鍵暗号と公開鍵暗号

この暗号方式を大別しますと、共通鍵暗号と公開鍵暗号があります。御存じの方もたくさんいらっしゃると思いますけれども、共通鍵暗号は先ほどの絵で描きました暗号鍵と復号鍵が同じ物で、これを共通鍵とも呼びます。これの特徴としては、非常に処理が軽くて高速で動きますので、多量のデータを秘匿したり、あるいは一部認証などで使ったりします。具体的な例として幾つか方式を挙げておりますが、特に有名なのがＡＥＳです。これはＮＩＳＴ（米国標準技術研究所）が標準化した暗号で、デファクトスタンダードになっていると思います。

一方で、もう１つ公開鍵暗号というものがあります。こちらは暗号鍵と復号鍵が異なり、暗号鍵は公開するというものです。これは後ほどもう少しイメージ的に説明いたしますが、特徴としては共通鍵暗号に比べて処理が遅い。ただ、用途がいろいろありまして、お互いに鍵を共有したり電子署名や認証にも使われております。具体例として一番有名なのがＲＳＡ暗号と呼ばれるもので、これは電子署名の方式にも使われております。(図3)

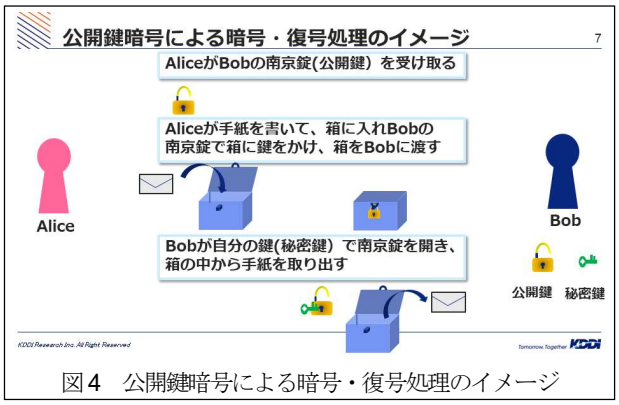

図４　公開鍵暗号による暗号・復号処理のイメージ

公開鍵暗号は初めて聞く方はイメージしづらいと思いますので、ここにちょっと絵を描いてみました(図4)。Alice さんと Bob さんという人の間で暗号通信をします。Alice さんから Bob さんに暗号文を送りたいというときにどうするかというと、この Bob さんのところに公開鍵と秘密鍵があります。公開鍵は南京錠を想定していただければいいと思います。Bob さんはまず Alice さんに南京錠を送ります。Alice さんは何か秘密の手紙を書いてこの箱の中に入れて南京錠で鍵をかけます。南京錠は開いていますので、誰でも鍵がかけられます。これを Bob さんに送ります。そうするとこの鍵をかけられた南京錠を解けるのは、それに対応する秘密鍵を持っている Bob さんだけなので、Bob さんはこの南京錠の秘密鍵を使って箱を開けて中の電文を読むことができるという仕組みになります。

これは何を言っているかというと、この南京錠＝公開鍵を使えば誰でも暗号化できます。ただ、その暗号化を解ける人はその南京錠の公開鍵に対応する秘密鍵を持っている人、すなわち Bob さんだけです。誰でも暗号化はできるけれども解けるのは本人だけですよという仕組みになります。こういった仕組みが非常に一般的によく使われています。特にインターネットのような不特定多数の人と暗号通信をするときに非常に有効だと言われています。

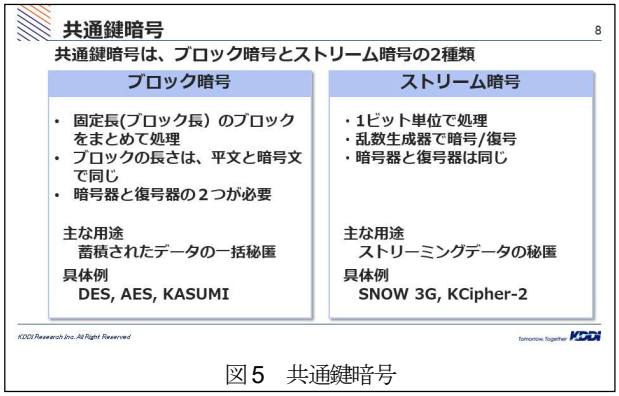

図５　共通鍵暗号

公開鍵暗号の話は最後の部分でも出てきますので話を共通鍵暗号に戻しますけれども、この共通鍵暗号はさらにブロック暗号とストリーム暗号に分かれます。この説明は非常に直感的で厳密ではないのですが、イメージとして捉えていただくために説明します。(図5)

ブロック暗号というのは暗号化する文のブロックを適当な長さにまとめて処理するというものになります。暗号したり復号したりするので、それの暗号器と復号器との２つがあります。主な用途としては、大きなデータを一括して暗号化するのに使います。

ストリーム暗号というのはビット単位で処理します。これも後ほど詳しく説明しますが、乱数生成器を使って暗号、復号をします。主な用途としては、ストリームデータ、要はリアルタイムの音声や映像など流れているデータを暗号化するのに適していると言われています。

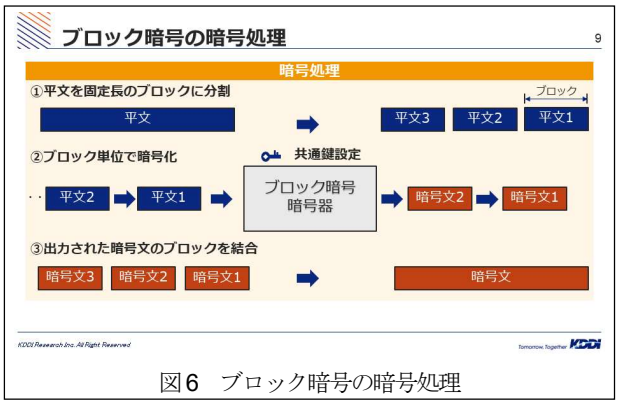

図6　ブロック暗号の暗号処理

今、説明しましたブロック暗号のイメージは、まず平文を適当な長さの固定のブロックに分けまして、それを１個１個ブロック暗号の暗号器にかけると、ブロックごとに暗号文が出てきます。出てきた暗号文をもう一度つなぎ合わせると暗号文が出来上がります。(図6)

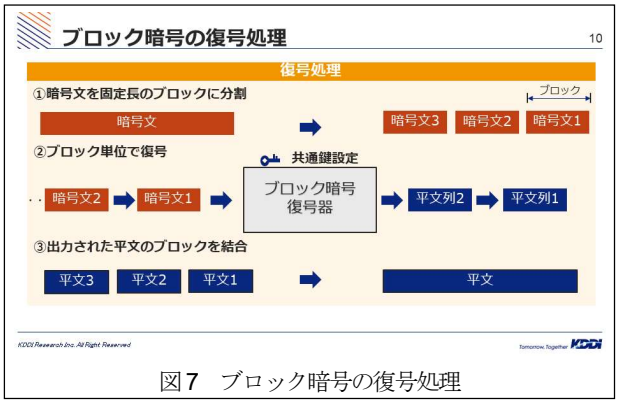

図7　ブロック暗号の復号処理

これと全く逆のことを復号処理でやるわけです。暗号文をブロックに分割して１個１個ブロックで復号化して、それをまたつなぎ合わせると平文が出てきますという流れになります。(図7)

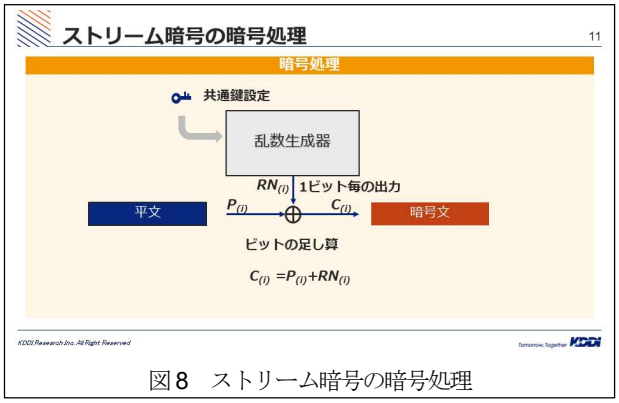

図8　ストリーム暗号の暗号処理

ストリーム暗号というのは少し動きが違いまして、０、１のビット列をランダムに出力する乱数生成器を用います。もちろん共通鍵は設定する必要がありますが、乱数生成器から出てきた乱数と暗号化したい平文をビットの足し算をします。これが暗号文になります。(図8)

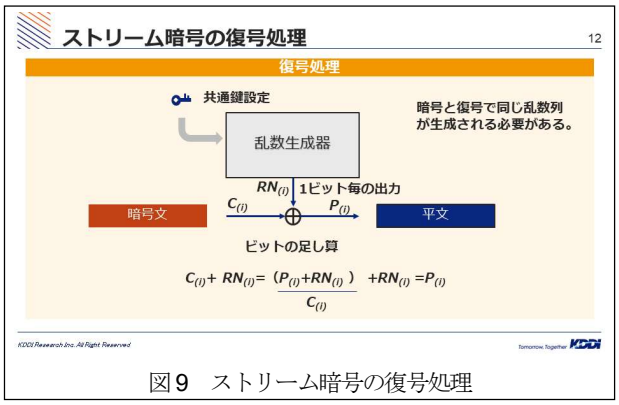

図9　ストリーム暗号の復号処理

復号はどうしますかということになりますが、構造は全く同じです。今度は暗号文を乱数生成器で出てきた乱数と足し合わせます。そうすると平文に戻りますということです。

これはどういうことかというと、もともと暗号文は平文（P(i)）と乱数（RN(i)）を足し合わせたものでした。それにもう一度乱数（RN(i)）を足し合わせるということになります。この足し合わせた乱数が元の平文を暗号化したときの乱数と同じものであれば、ここはビットの足し算になると消えてしまいます。そして元の平文（P(i)）が浮き出てくるという形になります。この暗号の特徴としては、暗号化するときの乱数列と復号化するときの乱数列が同じ乱数でないとこのように相殺されませんので、この乱数生成器は同期している必要があります。そこでこの方式は同期型のストリーム暗号と呼ばれております。(図9)

これ以外に自分で同期するようなもの、自己同期型ストリーム暗号もありますが、ここでは時間の関係で割愛させていただきます。

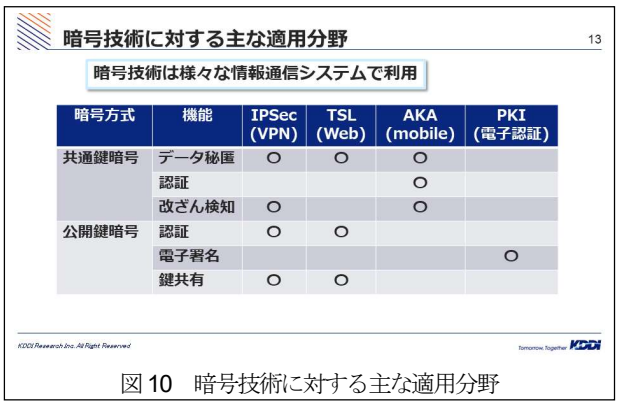

図 10　暗号技術に対する主な適用分野

例えば実際に使われているものとして、ＶＰＮは皆さん御存じだと思いますが、ＶＰＮの中にＩＰＳｅｃというプロトコルの仕様がありますけれども、ここでは共通鍵暗号でデータ秘匿、改ざん検知、公開鍵暗号では認証や鍵共有といった仕組みが使われております。ＷｅｂはＴＳＬという暗号プロトコルが使われております。この辺はインターネット系ですけれども、あとｍｏｂｉｌｅですね、移動通信系ですと皆さんも使っているスマホをネットワークにアクセスするときにキャリアが認証したり、無線の区間を暗号化するという機能がありますけれども、それはもともと携帯から来ているので非常に軽い処理が必要だということで主に共通鍵暗号でデータ秘匿、認証、改ざん検知をやっています。あとはマイナンバーカードに使われている電子認証も公開鍵暗号の電子署名が使われているということで、皆さんこういった暗号技術は普段から使われているということは認識していただけると思います。(図 10)

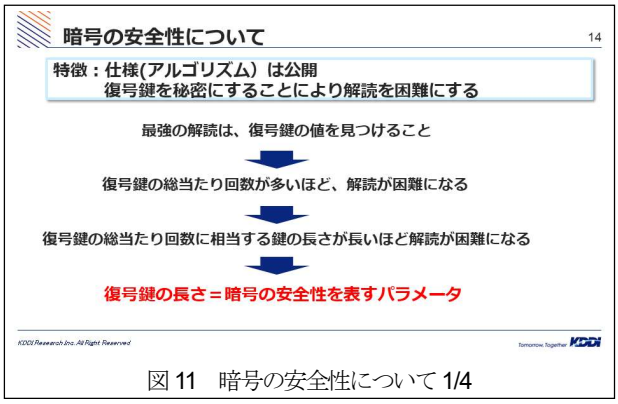

図 11　暗号の安全性について 1/4

では、暗号はどういったことを根拠に安全だと言っているかということについて、ここで少し説明させていただきます。先ほど現代暗号は仕様が公開されていて鍵を秘密にして安全性を担保していると言いました。ということなので、この鍵を見つけられると暗号が解かれてしまいます。最強の解読は鍵を見つけることです。鍵はあるビットの長さになっていまして、例えば短ければ、１ビットでは２通り、２ビットでは４通りとなりますから、２ビットであれば４回やればどれか鍵が当たることになるので、この復号鍵で総当たりをやることが１つの解読になります。当然ながら総当たりの回数が多いほど解読が困難になりますので、いわゆる復号鍵のビットの長さが長いほど解読が難しいということになり、安全性を表すパラメータとしては復号鍵の長さが１つの指標になります。(図 11)

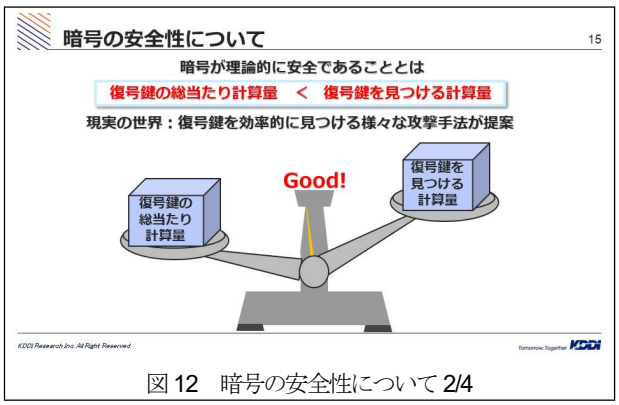

図 12　暗号の安全性について 2/4

何を言いたいかというと、復号鍵の総当たりの計算量が一生懸命やって暗号が解ける計算量ですね。もし復号鍵が、本当は総当たりが 100 通りだけれども 50 通りあれば見つかるような方法ができれば、これは「この暗号は弱い」と言われてしまいます。総当たりよりも復号鍵を見つける効率的な方法があれば弱い。逆に言うと総当たりよりも復号鍵を見つける効率的な方法がない、総当たりがやはり一番効率的だということになると、この暗号は弱点がないと言われています(図 12)。

では実際の世の中はどうなっているかというと、いろいろな暗号が提案されているのですが、やはり復号鍵を効率的に見つけるようなさまざまな攻撃の手法が提案されています。

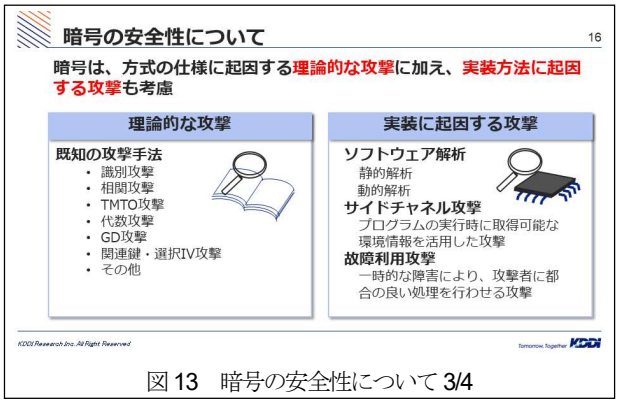

図 13　暗号の安全性について 3/4

今、申し上げましたのが鍵を推定する方法です(図 13)。いろいろな効率的な方法が攻撃ということで提案されていますけれども、これはあくまでも暗号が公開されていて仕様が分かっている中で、紙と鉛筆でひょっとしたらこうやってこうすると、非常にショートカットでこういう効率的な方法があるということを一生懸命研究したり、あるいは攻撃者が試みます。そういったものは理論的な攻撃と言われています。

ストリーム暗号の例で言うと、これは細かいので１個１個は説明しませんけれども、いろいろ攻撃が知られています。一方で、そういう理論的な攻撃だけではなく、暗号というのは、ソフトウェアあるいは電子回路で実際に物を作るわけですが、その作った物に対してその脆弱性、方式はちゃんとしているけれども実装のやり方が悪いので何か問題が起きるということがあります。これは実装に起因する攻撃と言われています。

これをやる手法としては、例えばソフトウェアを解析、逆アセンブルしたり、あるいは実際のソフトウェアを動かしていろいろなところの変数を見るなど、静的解析、動的解析ということをやることもありますし、サイドチャネルと言ってプログラ

ム実行時に取得可能な環境情報、例えばＣＰＵのキャッシュの値だとか、あるいはＣＰＵの消費電力から鍵を特定する仕組みも考案されております。また、故障利用といって、１カ所わざと故障させて動作させることによってデータが漏洩してしまう等、実際に物を作ることによって、実装に起因する脆弱性を攻撃する方法もあります。

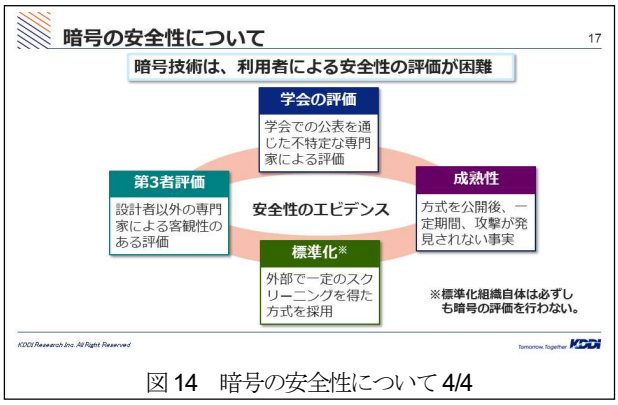

図 14　暗号の安全性について 4/4

ということで暗号の攻撃には理論的な攻撃もありますし、実装に起因する攻撃もありますが、実際に使う人の側にとってみると、かなり専門的過ぎて分からないです。例えば、暗号を提案するベンダーさんが「これは安全です」と言っても、本当に安全なのか分からない。そうすると、使う側にとってみれば何をもって安全と担保を取ればいいのだろう、安全性のエビデンスは何だろうという非常に難しい議論になってきます。

世の中一般的に言われているのはこういうことだと思います。１つは、暗号は先ほど言いましたように公開します。例えば学会で仕様を公表します。そうするとそれに対して不特定多数の研究者が攻撃するので、不特定な専門家による評価が行われます。

また、第三者評価といって、自分たちが作った暗号を誰かにお願いして評価してもらうといったようなことで客観性のある評価をすることもあります。こういった学会の評価や第三者評価を通じて、例えば３年たってもどうも攻撃が見つからないということになると、これはひょっとしたら安全なのではないのというコンセンサスが出来上がってきます。これは一定期間攻撃が発見されないという事実をもってこの暗号は成熟しているといったような考えになります。

こういったものを総合的に考えていく上で、一番使われていて安心材料は、どこかの国際標準に標準化されている、あるいは国内で標準化されているというものがあれば、これは安心して使えるよね、というようなことが言えるのではないかと思っています。

ただ、気をつけないといけないのは、標準化機関というのはそもそも評価を行う組織ではないです。ただ、外部で一定のスクリーニング、評価を得たものを標準化するといったようなプロセスになっていますので、ある種の安全性の担保するプロセスが標準化とも言えます。(図 14)

ＫＣｉｐｈｅｒ-2

今御説明しました暗号の概要を踏まえて、私たちＫＣｉｐｈｅｒ-２はどういう開発の経緯で何をやってきたかというところをこれから御説明したいと思います。

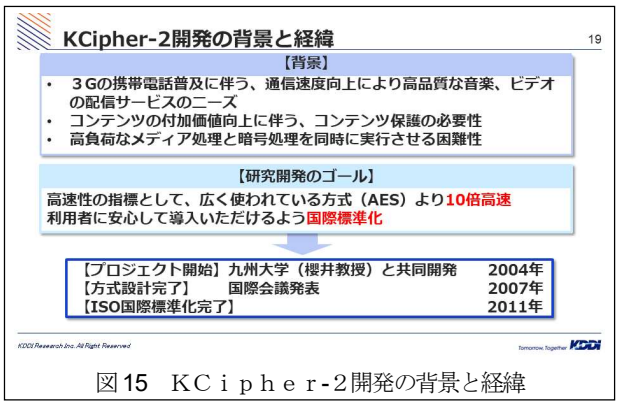

図 15　ＫＣｉｐｈｅｒ-２開発の背景と経緯

開発の背景と経緯ということで、冒頭で少し申し上げましたけれども、この技術は十数年前に開発を始めたものです。ここに書いてありますようにプロジェクトを開始したのは 2004 年です。当時、何をもって我々はこういうことを始めたかということですけれども、先ほど言いましたＡＥＳという米国のいわゆる業界標準の暗号が既にありましたし、そういう中で我々があえてキャリアとして暗号を作る意味は何なのかというところを少し御説明したいと思います。

当時は３Ｇの携帯が非常に普及してきておりました。今と比べると非常に稚拙ですけれども、当時は通信速度も非常に向上してきておりまして、それを使って高品質な音楽、ビデオを配信するサービスのニーズが非常に高まって、着うたやビデオの配信が立ち上がってきておりました。そういう中で流れるコンテンツが非常に価値が高い、例えば映像ですとハリウッドの映画が流れるといったようなサービスも出てくるわけです。そうすると当然、コンテンツを不正に複製するというリスクが発生しますので、コンテンツの保護の必要性が出てきました。

ただ、当時はこういった高負荷なメディア処理と暗号といったセキュリティーの処理を、例えば携帯電話で同時に実行させると結構重たい処理でした。従来のＡＥＳという業界標準的なものではうまく動かないといったことが分かってきました。ということで、私たちの研究開発としては、まずＡＥＳという今まで使っているものよりももっと速い、目標としては１桁速いということを目標に置きました。こういったものを作って、携帯電話等でも高付加価値のコンテンツをさくさくと暗号処理できるようなものを世の中に出したいということが１つのゴールとなりました。

ただ、コンテンツホルダー、いわゆるハリウッドさんや音楽業界さんが、例えば携帯でそういった高品質なコンテンツを流そうといったときに、使われている技術は本当に安全ですか、ということが問われるわけです。そういったことに応える意味で利用者に安心していただけるように国際標準化がやはり必要だろうということで、この 10 倍高速と国際標準化を目標として研究プロジェクトを始めました。

先ほど言いましたように 2004 年から始めて、方式が完了したのが2007年で、これは実際に国際会議で発表しております。ここから第２の目標である標準化を行いまして、約４年半かけてＩＳＯの国際標準化が完了しております。(図 15)

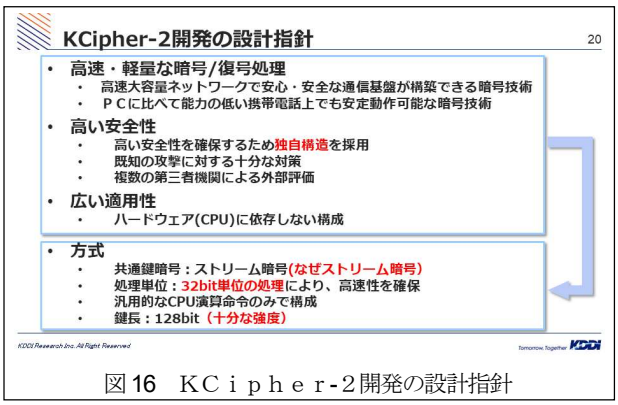

図16　ＫＣｉｐｈｅｒ-２開発の設計指針

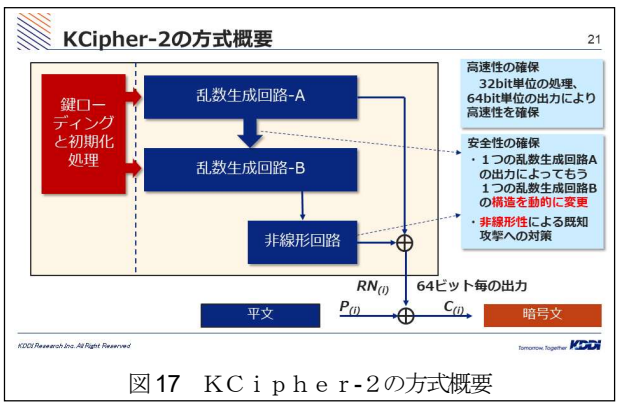

図17　ＫＣｉｐｈｅｒ-２の方式概要

以上が開発の背景ですけれども、では具体的にどういう設計指針でやったかということになります。先ほど言いましたように１つは高速・軽量で動くものということを大きな目標としてきました。２番目は高い安全性です。具体的な設計指針としては、安全性を向上させるため独自構造を取り入れ、一般的なストリーム暗号の攻撃に対する十分な対策を行うとともに、複数の外部の評価機関に第３者評価を依頼しました。

独自構造の採用についてはある種リスクがある話で、暗号というのはある程度お決まりの作法のようなもの、特にブロック暗号はできていまして、それで作ればある程度しっかりとしたものができるということが、だんだん研究が進んで分かってきていましたが、開発当時ストリーム暗号はまだそこまで技術的に確立されていなかった。なおかつ、そこに独自構造を入れるということは、それはひょっとしたら独自構造を入れることによってセキュリティーが上がるかもしれないですし、あるいは新しい技術を入れることで脆弱になるかもしれない。ここは１つ大きなリスクでした。ただ、我々はそういう意味ではリスクを取って設計をしました。

あと、ハードウェアに依存しないということで、ＣＰＵのいろいろな特性がありますから、そういったものに依存しないような構成を考えまして、いろいろなところに使っていただくようにしました。

もう少し具体的な指標としましては、今言いましたストリーム暗号を使って高速に処理するために、先ほど冒頭でストリーム暗号は１bit処理をすると言いましたが、それをさらに32bit、いわゆるＣＰＵのワード単位の処理で高速性を確保しました。汎用的なＣＰＵの演算命令を使うということで、ハードウェアやＣＰＵに依存しない、いろいろな広いプラットホームで使っていただけるような暗号としました。鍵長は当時、十分な安全性が保証されているだろうと言われていた128bitの鍵長を使っています。

当時はブロック暗号がかなり主流だったわけですけれども、我々はストリーム暗号を選択しました。これは先ほど申し上げましたようにブロック暗号はかなりかっちりと学問としても成り立っていて、できたものもかなり安全性が高い。ただ、処理がストリーム暗号と比較して遅いというデメリットがありました。私たちは、それは少しリスクにはなりますけれども、軽量なストリーム暗号という方式を採用しました。(図16)

具体的にどういうものかということですけれども、ここからはかなり技術的な話になるので、できるだけ簡単に説明しようと思います。ストリーム暗号というのは乱数生成器で構成されますが、我々は乱数生成器を２つ作りました。そこに非線形回路――ここは詳しくは説明しませんけれども、こういった回路を合わせた構成にしております。先ほど言いました 32bit 単位で乱数生成をするので、処理が非常に高速になります。なおかつ出力は32bit 合わせて 64bit とし、１回ＣＰＵのクロックが回るごとに 64bit 出てくるということで、従来のストリーム暗号よりも随分高速な設計をしました。

安全性を確保するという意味で、これは後ほど詳しく説明いたしますが、先ほど言いました「独自の構造」を導入しております。これは動的に回路を変える仕組みです。もう１つは、既存のいろいろな攻撃に耐性があるように非線形回路を導入しております。(図17)

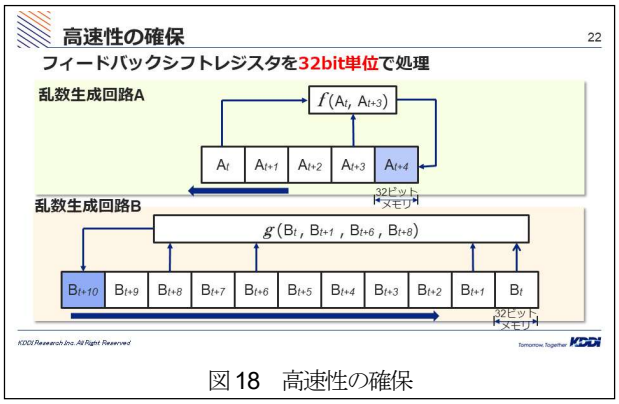

図18　高速性の確保

高速性とはどういったものかということですけれども、乱数生成器Aは 32bit ごとのメモリになっていまして、これが１回クロックを回るごとに32bit、右から左にシフトします。そうすると一番右側は何も入力がありませんので最初の 32bit のメモリのところからある関数の出力結果を入力する。これをぐるぐる回すというものです。これはフィードバックシフトレジスタと呼ばれています。同じように乱数生成器Ｂもフィードバックシフトレジスタを使って 32bit ごとに処理をするということで高速性を担保しています。(図18)

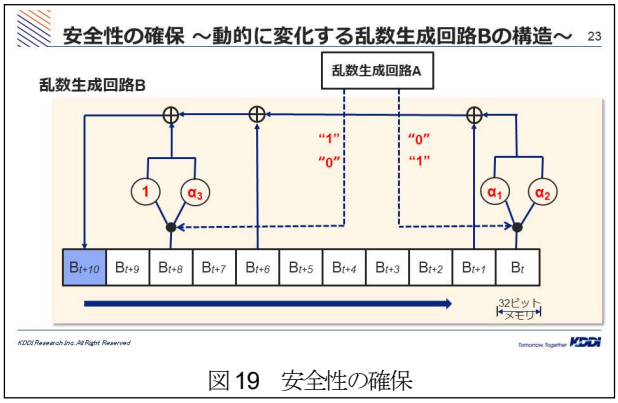

図19　安全性の確保

安全性を担保するための動的な構造ということですけれども、これは乱数生成器Aから出てきた出力の一部を取り出しまして、この値によってもう一方の乱数生成器Bのフィードバックをするときの関数の構造を動的に変更させます。例えば、乱数生成器Aの左側の出力の値が“0”、右側の出力の値が“1”(これを0・1と表す)の場合ですと、動的な関数の左側が“1”と右側が“α2”を選んで計算します。逆に、例えば1・0の場合には“α1”と“α3”が選択されます。このように0・1、1・0、1・1、0・0と4通りありますが、この4通りの回路構造を関数の出力によって毎回、毎回、変えるということで、非常に解析を複雑にするようなことをしています。これが独自構造と言われるものでして、これが本当に安全なのかどうかというところですけれども、我々が机上で評価したところによるとやはり従来の固定的な構造を使うよりは動的な構造を使うことによって非常に解析が難しくなるということが分かってきました。(図19)

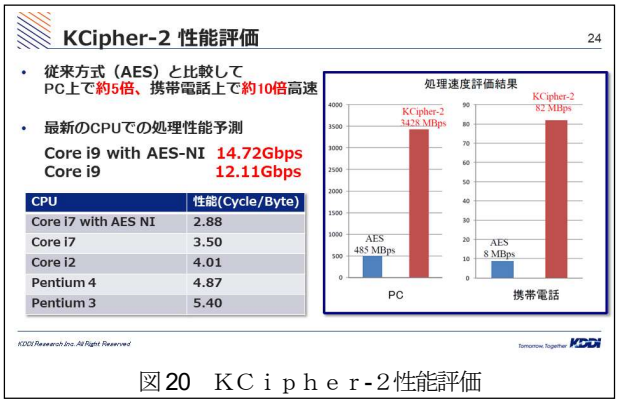

図20　ＫＣｉｐｈｅｒ-2性能評価

こういった構造を実装して実測しました(図20)。当初、目標としていたＡＥＳという業界標準的な方式と比べて、ＰＣ上では10倍はいきませんでしたけれども5倍、携帯電話上では目標通り約10倍の高速性を確保できました。

例えば今のＣｏｒｅ　ｉ９という最新のＣＰＵで動かしてみると大体見積もりができていまして、特殊なハードウェアがなければ12Gbpsぐらいで、ある特殊な演算命令を使うと15Gbpsぐらい出るだろうということが分かっております。これぐらい出ると、少なくとも今の例えば５Ｇの携帯では速度として十分使えるような仕様になっております。

ではＫＣｉｐｈｅｒ-2の安全性についてどうなのかということも当然いろいろ検討しておりまして、これは再掲(図13)になりますけれども、理論的な攻撃、それから実装に起因する攻撃、それぞれ検証をしております。

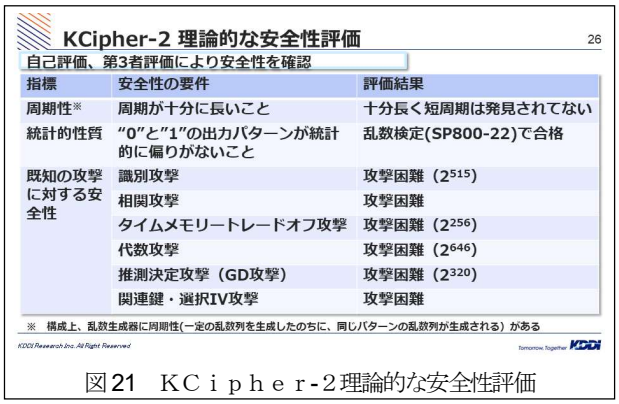

図21　ＫＣｉｐｈｅｒ-2理論的な安全性評価

これは少し細かくなり過ぎるので御説明しませんが、理論的な安全性の評価として３つあります。周期性――同じビット列を繰り返すことが構造上ありますが、それが短いとやはり弱いということになりますので、十分に周期が長いこと。

それから乱数生成器ですので、０・１の出力パターンが統計的に偏りがないこと、これは基本的な安全性を確保するために必要な評価になります。そして、既存のいろいろ知られている攻撃に対して十分安全であること、これは理論的な評価になりますけれども、こちらも評価の結果、特に問題がないということが、我々の自己評価だけではなく第三者にも評価していただいて、分かってきております。(図21)

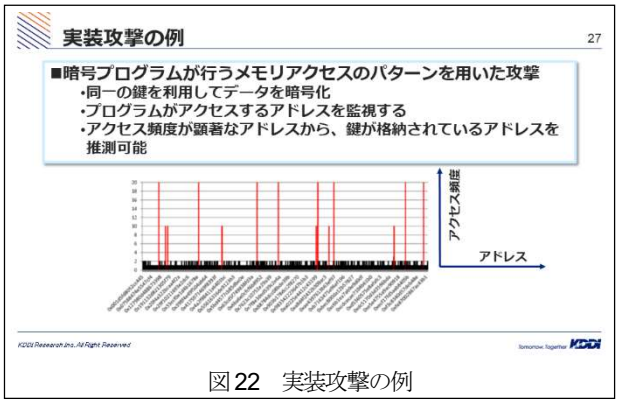

図22　実装攻撃の例

一方、実装の安全性につきましてはいろいろ観点がありますけれども、ここには１つ例として挙げさせていただきました。メモリのアクセスパターンを使った攻撃がございます。どういうことかというと、暗号というのは先ほども言いましたソフトウェアで実装する場合がありますので、実際に動かすときにマシーンランゲージに落ちて、それが計算機のメモリ上に展開されてそれを１個１個メモリを実行していくことで暗号化の処理ができるわけです。

そうしますと、その中で暗号というのは鍵を設定しますから、ある鍵があるメモリ上に置かれます。暗号化するときにはその鍵を参照しながら処理をするわけですから、例えば同じ鍵をずっと使っていると、常に同じところのメモリアドレス上をアクセスすることが何となく分かってきます。そうしますとこのプログラムがアクセスするアドレスをずっと監視していると、どうもアドレスのメモリが毎回、毎回アクセスしているなということで、鍵が格納されているアドレスではないかということが

分かってきます。

鍵の総当たりですと、例えばすごい回数やらなければいけないですけれども、これでメモリが格納されているアドレスがある程度絞られると総当たりの回数がぐっと減っていくわけですから、効率的な解読ができることになります。したがって、鍵が格納されているアドレスを推測されてしまうことが非常に問題だということになります。これは幾ら方式の仕様が安全であっても、ソフトウェアで動かすことによって問題が発生するという意味です。(図 22)

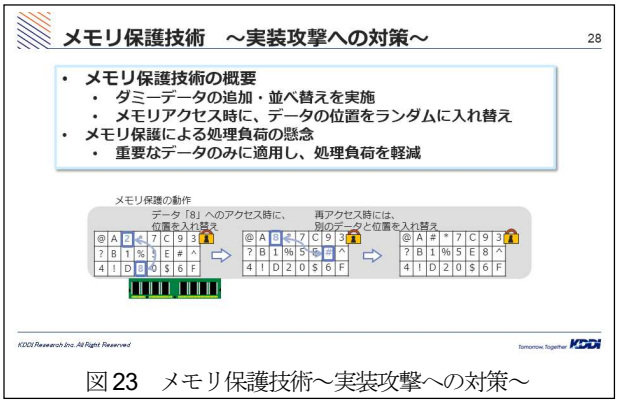

図 23　メモリ保護技術～実装攻撃への対策～

例えばこれを防ぐ方法として私たちはメモリ保護技術を開発しております。これは端的に言いますと、先ほどの固定的なところをアクセスされないようにダミーデータを入れて目くらましをするといったようなことと、あとはメモリにアクセスしたときに、今度アクセスするときにはデータを別のところに動かしてしまい、データの位置をランダムに入れ替えることによって、暗号鍵のような重要なデータが常に同じメモリ上の固定のアドレスにいないようにする技術を使って、先ほどのような攻撃を回避する仕組みになっています。

ただ、これは処理が非常に重くなりまして、これをやることによって処理負荷が大きくなるという懸念があります。ただ、これはプログラム全体にこういう処理を施すわけではなく、暗号化するのに必要な鍵のデータをアクセスするところに対してのみこういう対策を施すことによって、処理負荷は軽減できることが分かってきております。(図 23)

今、言いました安全設計の指針や安全性の評価をやってきておりましたけれども、もう１つの目標であります標準化について、少し苦労話も含めてお話しさせていただきます。

図 24　ＫＣｉｐｈｅｒ-２の標準化

私たちの方式は国際標準機関のＩＳＯの標準と国内の電子政府推奨暗号ＣＲＹＰＴＲＥＣ、それからインターネット標準のＲＦＣに展開しました。この中でも特にＩＳＯの活動について少し御紹介したいと思います。(図 24)

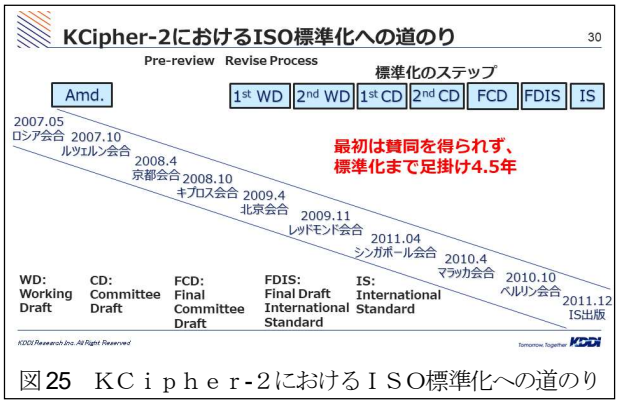

図 25　ＫＣｉｐｈｅｒ-２におけるＩＳＯ標準化への道のり

ＩＳＯでは標準化するに当たって非常に長いステップがかかります。まず、WDというワーキングドラフトから始まりコミッティードラフト、ファイナルコミッティードラフト、ＤＩＳ(ドラフトインターナショナルスタンダード)、ファイナルドラフトインターナショナルスタンダード、その後ＩＳというのが通常のステップです。非常に長いステップになりますので、これを飽きずにずっと提案し続けないと標準化されないということになります。しかも私たちが最初に提案したときに、ここにアメンドメント（Ａｍｄ.）と書いてありますけれども、既存の方式に１個加えるという、プラスαで新しいアルゴリズムを入れるという形で提案しましたが、最初のロシア会合で、ものの見事に否決されました。実際、ここから４回ぐらい会合をやり、やっと 2008 年キプロス会合でスタンダードの見直しをやろうというのでワーキングドラフトを新しく作ることになりまして、そこで初めて我々の方式が暫定的に入れてもいいよということになりました。

なぜずっと否決され続けたかということですけれども、先ほど暗号というのは成熟性ということを言いましたけれども、どうも新しい暗号は信用されないわけです。やはりある程度年月がたち、どこからも攻撃されていないという事実によってはじめて、どうも「この方式は安全らしい」と思われるわけです。一般的にこの信用を得る期間は３～５年と言われています。我々が提案してから大体２年ぐらいたってやっとドラフトの中に入れていただいたという経緯になっています。これは精神的にもつらい思いをしました。(図 25)

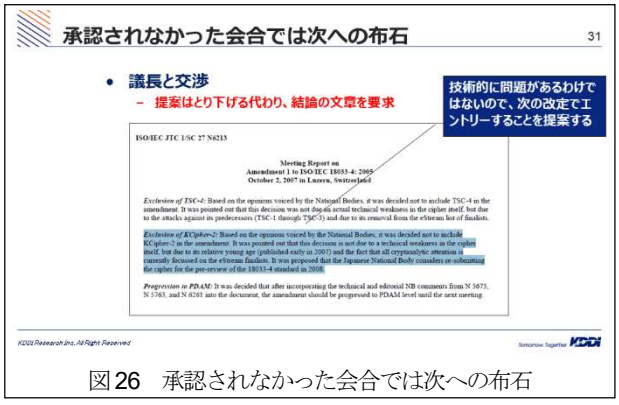

図 26　承認されなかった会合では次への布石

その１つの例ですけれども、これは最初のアメンドメントという、ちょっと入れてくださいというところで断られたときのレゾリューションです。我々はこのレゾリューションの文面に非常にこだわりました。これは当時の議長に「君たちの方式はだめだよ、新しいから信用できないよ」と言われて、「それは分かった、取り下げるのはいいけれども、こういう一文を入れてくれ」と言いました。「これは技術的に問題があるわけではない、ただ単に新しいからなのだ、だから次の改定でエントリーすることを提案したらどうか」ということをぜひこの文面に入れてくれということをやりながら、何とか自分たちの方式を採用してもらう努力をしました。

なぜこういうことを一生懸命やったかというと、先ほど言いましたように標準化は使っていただく方々、お客様にとって１つの指標になるからです。実を申し上げますと我々が標準化をやっている最中からいろいろなお客様にこういう提案をしてきました。そういうときの売り文句として「これは今、標準化をやっていまして、必ず標準化されますから」というセールストークを言いながらやっているわけですから、それで否決されたら我々は会議から日本に戻ってくることができないという覚悟でやらざるを得なかったという事情もあります。標準化を担当されている方、特に会社の方式などをやられている方は皆さんそうだと思いますが、会社を背負ってやっているというところで、本当に我々としても非常に苦労したところです。

こういうこともありまして、あとは皆さんの協力もあって、標準化がされました。これはＩＳＯの 18033-4 ストリーム暗号の中に我々の方式を入れていただいたということになります。

あとはＣＲＹＰＴＲＥＣという電子政府推奨暗号リストのストリーム暗号のカテゴリーの中に、私どもが提案したＫＣｉｐｈｅｒ-２もＣＲＹＰＴＲＥＣの中で評価した上で入れていただいているという状況になっております。(図 26)

暗号ですのでいろいろな分野に使われるのは当然ですけれども、いろいろなところに展開させていただいております。当初、開発を始めた頃は携帯電話で既存の方式は処理が重いということで始めましたが、標準化するのに４～５年かかっているうちに、あれよあれよという間に世の中はスマホになりまして、ＰＣと同じぐらいの処理速度になってきたということで、そんなに軽いものは要らないのではないか、という時代にもなってきました。

ただ、そうなってきたとしても、やはり軽い暗号はニーズがあります。１個１個の説明は省きますけれども、最近では特にタグ型のビーコン、いわゆるＩｏＴ関係の分野、あるいは監視カメラなどには結構いろいろニーズがあって、お客様からお問合せをいただいているという状況になります。

あとは自社でもいろいろ製品として使っております。ストリーム暗号自身はいわゆる動画のストリームデータや音声に対しての暗号に非常に向いているということなので、我々の製品でＶｉｓｔａＦｉｎｄｅｒは動画をスマホで配信して遠隔から監視して作業を支援するといったシステムですけれども、ここの暗号化の技術として使っております。

最近ですと４Ｋなど高精細な映像を使って作業支援をしたいというニーズも出てきておりますので、そうするとスマホやマシンがどんどん進化して計算能力も速くなってきておりますけれども、やはり暗号は軽いほうがいいというニーズがありまして、こういった我々の方式を使っているというような形になります。(図 27)

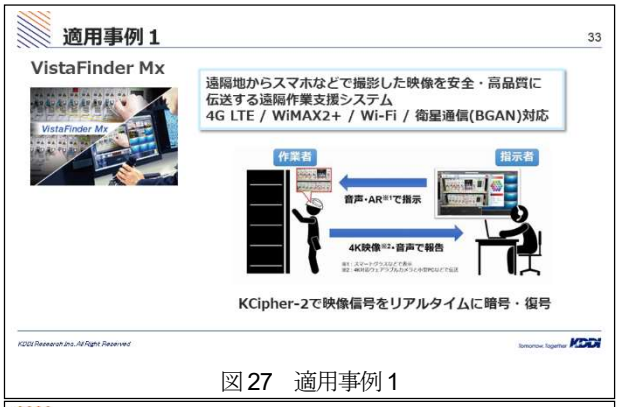

図 27　適用事例１

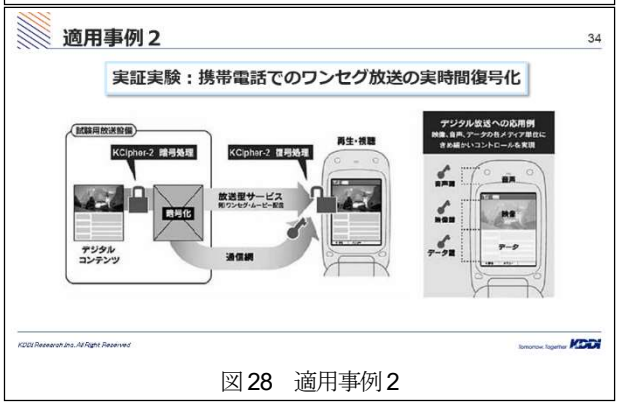

図 28　適用事例２

あとは古いものですけれども、開発した当時、携帯電話でワンセグ放送が出てきました。ワンセグで付加価値コンテンツを流すときに暗号化するといった話も、当時、ニーズとしてありまして、放送型サービスのデータをＫＣｉｐｈｅｒ-２を使って暗号化するという実証実験もやっております。(図 28)

暗号技術の今後

今回受賞しましたＫＣｉｐｈｅｒ-２の開発の経緯、それから標準化の苦労話、どういったところに使われているかといったことを御説明させていただきました。ここからは少し受賞の枠から外れますけれども、今、世の中の暗号技術、トレンドとはどういったものかということと、それに関わる私どもの取り組みなども御説明させていただきたいと思います。

図 29　量子コンピュータ技術の進展

最近、量子コンピュータが非常に話題になってきております。従来のスパコンの性能をはるかにしのぐ量子コンピュータがかなり現実的になってきたと言われています。実はまだまだ実現は遠いのではないかという話もありますけれども、目的限定といった形であればかなり実用的になってくるということになります。

量子コンピュータは大きく分けて２つあります。１つはゲート型量子コンピュータ、これは汎用マシンです。スパコンに代わるものになります。去年でしたか、Google 社が「量子超越性」を実証した、特定の計算でスーパーコンピュータを超える処理能力ができたというようなことを言っております。一方で最適化問題に特化した量子コンピュータがあります。これは量子アニーリングマシンですが、こういったものも今、国内外でいろいろ研究が進んでいます。(図 29)

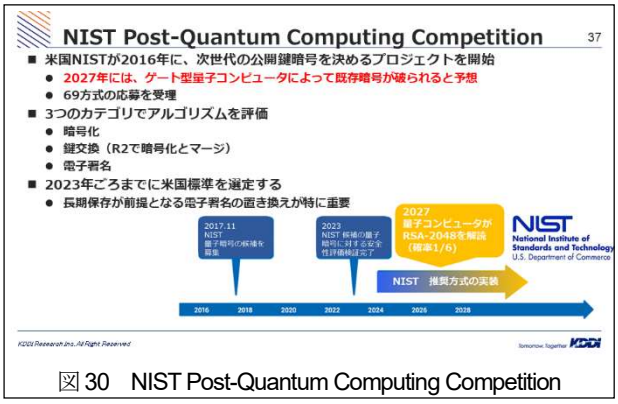

図 30　NIST Post-Quantum Computing Competition

こういった流れの中で、従来使っていた暗号がどうも危なくなってきているぞということが言われています。今日のメインは共通鍵暗号、ストリーム暗号、ＫＣｉｐｈｅｒ-２でしたが、もう１つの公開鍵暗号はゲート型の量子コンピュータによって効率的に解かれるアルゴリズムが発見されておりまして、ＲＳＡ暗号、公開鍵暗号はどうも危ない、2027 年ぐらいにはゲート型量子コンピュータが実現されそうだということで、2027 年には公開鍵暗号が使えなくなる、今世の中で一番使われている、インターネット等で使われている暗号が使えなくなるといったようなことが予測されております。

こういった予測を踏まえて、米国の標準機関ＮＩＳＴは 2016 年から次世代の公開鍵暗号を決めるプロジェクトを開始しております。実際、今は１次応募、２次応募というところぐらいまで終わっていますが、１次応募では 69 方式の応募がありました。具体的なカテゴリーとしては暗号化、鍵交換――前半では鍵共有と申し上げましたが、鍵共有、鍵交換、そして電子署名といったアルゴリズムが提案されています。

2027 年にゲート型の量子コンピュータでＲＳＡ暗号が破られるのではないかという予想に基づいて、では標準化はいつ頃すればいいのかということで、2027 年に破られるから 2027 年に出来上がればいいというわけではなく、やはり方式ができて実用化するまで３～４年かかりますということで、ＮＩＳＴでは 2023 年頃までに米国の標準を作りましょうといったような形で動きが出てきております。(図 30)

図 31　耐量子暗号を実現する方式

これは先ほどのお話と少しかぶりますけれども、現代暗号は公開鍵暗号のＲＳＡ暗号で、素因数分解という数学の基礎問題が非常に難しいというようなことに基づいてこの暗号は成り立っています。素因数分解はゲート型の量子コンピュータによって非常に効率的に解けることが分かってきているということで、ＲＳＡ暗号が危ないと言われています。素因数分解というのは中学生でも習うような６の素因数分解、２×３のような話ではありますが、桁数が非常に大きくなるときわめて難しいと言われていまして、こういったような基礎問題をうまく使った暗号がＲＳＡ暗号になっております。

それが破られるとなると、それに代わるもの、量子計算機に耐えられる暗号ということで耐量子暗号というものがいろいろ検討されて始めています。これもＲＳＡ暗号と同じようにある種の安全性の根拠となる数学的な基礎問題をうまく使って方式が検討されております。具体的には細かいので内容については御説明しませんけれども、格子暗号、多変数多項式暗号、符号暗号はそれぞれ少しずつ基礎問題が違っておりまして、こういった違った基礎問題を使って異なる種類の技術に基づいた暗号方式が、いろいろと検討されているという状況になります。

比較で言いますと、耐量子暗号なので量子コンピュータができても解けないということが１つ大きなメリットではありますけれども、あとＲＳＡ暗号に比べて提案されている暗号は結構速いと言われております。数倍から数十倍ということもあると思いますが、欠点もあります。鍵の長さあるいは電子署名の長さが非常に長く、今のＲＳＡ暗号では 1024bit といったものですが、耐量子暗号の鍵長になると数十キロバイトと非常に長いものになって使い勝手が悪いので、安全性を落とさずにいかに短くしていくかということが大きな問題になってきます。(図 31)

図 32　KDDI 総合研究所における耐量子暗号の研究・開発

ここからは、私どもがやっている手前みそのお話をさせていただきます。先ほどのお話に少し戻りますが、私たちは耐量子暗号の中でも格子暗号という１つの暗号技術を使って検討を進めております。これを検討するに当たっては、３つの技術の上に積み上がっています。一番基礎的な技術は、格子暗号の基礎になっている基礎問題が本当に安全なのか、どこまで安全なのか、例えばどういった鍵の長さを使えば安全なのかといったところを解析する取り組みです。逆に言うと、どうやれば効率的に破られるかということを一生懸命問い詰めることで、ここまでやれば安全だということが分かるということで、安全性の解析を一生懸命やっています。

今言ったように基礎問題が安全だということが分かれば、それを使った暗号方式が安全だということになりますので、そういった暗号方式の検討も私どもは進めております。具体的に言うと電子署名の方式を検討しております。こういう暗号技術を使って、実際のサービスやアプリケーションも取り組みとして考えています。暗号方式が安全であればそれを使った暗号のアプリケーションも安全になる、これは全部信頼関係が成り立ってきますので、そういったようなものも検討しております。

こちらは先ほど高橋さんのほうから御講演のありました、例えば完全準同型暗号ですね。クラウド上で暗号化したままいろいろ処理するといった取り組みを私どもも検討しております。これらを簡単に御説明します。(図 32)

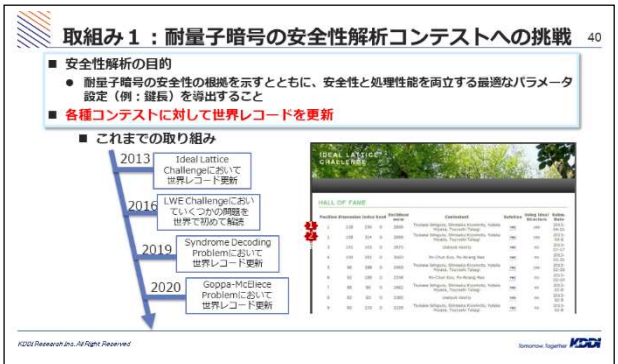

図 33 取組み１：耐量子暗号の安全性解析コンテストへの挑戦

まず、一番基礎的な数学問題の解析は、我々も安全性の解析コンテスト――ドイツのダルムシュタット工科大学がコンテストをやっていまして、いわゆるオリンピックのようなものですが、そこにいろいろチャレンジして研鑽をしているという状況です。「各種コンテストに対して世界レコードを更新」と書いてありますが、各種コンテストというのは、いわゆるオリンピックの競技種目のようなものです。陸上の 100 メートルでチャンピオンデータを取る人もいれば、マラソンでチャンピオンデータを取る人もいるという形で、いろいろなお題に向かって我々もトライをして、破られたりはしますが、世界レコードを更新したいということでいろいろな取り組みを進めています。(図 33)

図 34　取組み２：耐量子暗号　電子署名方式の設計

２番目は、そういった安全性の解析のもとに具体的な暗号方式の設計も進めておりまして、私たちは格子暗号を使った電子署名の検討を進めています。先ほど言いましたように何が問題かというと、やはり鍵長や電子署名の長さはまだ既存のＲＳＡ暗号より長く、使いづらいということですので、それをできるだけ安全性を落とさずに長さを削減するような仕組みを、一生懸命やっております。私たちはＬＷＲという基礎問題の上に署名方式を使っていまして、実際、最も有力だと言われているものに対して約 30%削減できたということで自慢をさせていただいているという状況になります。実用化にはまだまだ削減する必要があると思っておりますが、こういった取り組みをしております。(図 34)

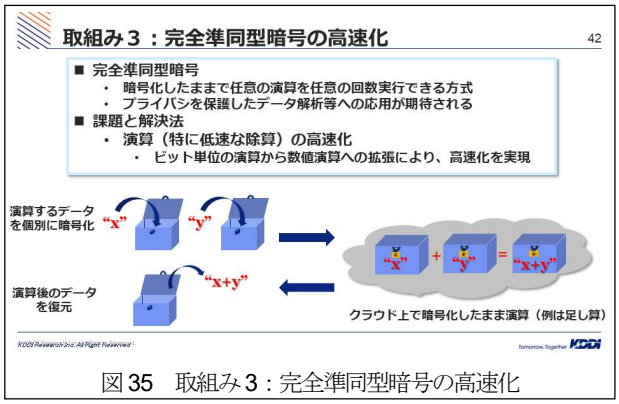

図 35　取組み３：完全準同型暗号の高速化

３番目は、こういう暗号方式を使った具体的なサービスの事例です。先ほど高橋さんから御講演がありましたプライバシーに関わりますけれども、完全準同型暗号についての検討もやっています。完全準同型暗号というのは、例えばデータをクラウド側で処理したい、でもクラウド側にデータを渡してしまうとクラウド上の人から見るとデータが見えてしまうということなので、暗号化したままデータを処理してもらいたいということを考えます。

例えばＸとＹの足し算をやりたいのですが、ＸとＹをそのまま足そうとするとクラウド上でこのＸとＹが見えてしまうので、一旦これを暗号化してそれをクラウド側に送って、暗号化されたものを足し算してもらう。それをもらって復号すると、足し算の結果が得られる。こういう状況ではクラウド上の人たちにはこのＸとＹとか、あるいは足し算をした結果、Ｘ＋Ｙは分からないということになりますので、プライバシー保護という意味でも非常に有効な手法だと言われております。

今は足し算の例で御説明しましたが、足し算については既に簡単な方式として提案されています。これをいろいろな任意の演算、あるいはプライバシーに関する統計の処理をやるとすると、完全な準同型暗号が必要になります。これについては一応、理論的には提案されております。ただし、非常に処理が重いという課題があります。

したがって暗号の研究者はいかにその処理を軽くするか、あるいは汎用的に軽くすることが難しい場合には、この計算については非常に速いという目的限定で高速化するような仕組みをいろいろ検討しているという状況になります。

私たちもそうした高速化の検討をやっております。特に割り算は非常に遅いことが分かっていまして、これの高速化の検討をいろいろ取り組んでいる状況になります。まだまだこの辺は研究段階になりますが、狙いとしては先ほど言いました耐量子暗号ですね、量子ゲート型のコンピュータができても耐えられる分野ということで、こういった取り組みを進めております。(図35)

むすび

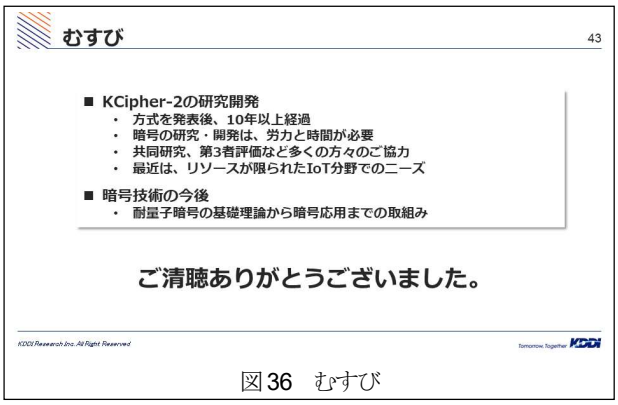

図36　むすび

そろそろ時間になりましたので、結びとさせていただきます。前半では暗号の概要に加えて、今回受賞させていただきましたＫＣｉｐｈｅｒ-２の研究開発の経緯を説明しました。先ほど言いましたように方式を発表してから 10 年以上経過しておりますけれども、幸いにも効率的な解読は発見されていないということかと思います。

ただ、暗号の研究は非常に労力と時間が必要です。先ほども言いましたアルゴリズムの設計に３～４年かかっています。それから標準化して皆さんに使っていただけるまでにまた３～４年ということで、非常に労力がかかるものです。ということで、単独ではなかなか難しいのかと思っておりまして、私たちも今回の共同研究、第三者評価、あるいは標準化で一緒に活動させていただいた国内の方々などいろいろな御協力を得てやっとこういうところまで来たなと思っております。

計算能力がどんどん発達してくると、先ほど言いましたようにスマホもＰＣと同じ能力なので軽量のものは要らないのではないかという考え方もありますが、やはり世の中にはいろいろ新しいニーズがありまして、最近ではＩｏＴ分野、特にスモールデバイスで軽量の方式に非常に高いニーズがあります。したがってこういう暗号の研究は常に進化していかなければいけない分野ではないかと思っております。

特に最近は耐量子暗号の取り組みが非常に重要になってきております。何とか私たちもこういったところに取り組んで、引き続き世の中に貢献させていただければと思っております。(図36)

時間になりましたので私の講演はこれで終わりたいと思います。御清聴どうもありがとうございました。

本講演録は、令和２年 10 月 27 日に開催された SCAT 主催「第 108 回テレコム技術情報セミナー」のテーマ、「Society5.0 時代における国民の安心安全を支える研究業績」の講演内容です。
＊掲載の記事・写真・イラストなど、すべてのコンテンツの無断複写・転載・公衆送信等を禁じます。